Welcome to Machine Learning using Python

- We begin by importing the necessary libraries to this dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- Let's introduce our data
- We obtained this from UCI Machine Learning Repository
- Information to this dataset is contained in the link below https://archive.ics.uci.edu/ml/datasets/Automobile

In [2]:
```
auto=pd.read_csv('auto.csv')
auto.columns=['symboling','normalized-losses','make','fuel-type','aspiration',
             'num-of-doors','body-style','drive-wheels','engine-location','wheel-base',
             'length','width','height','curb-weight','engine-type','num-of-cylinders',
             'engine-size','fuel-system','bore','stroke','compression-ratio','horsepower',
             'peak-rpm','city-mpg','highway-mpg','price']
auto.head()
```

Out[2]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepowe
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	11
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	11
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	15
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	10.0	10
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	8.0	11

5 rows × 26 columns

- As you see, we use pandas to activate csv files
- Also, auto.head() allows us to see a small portion of our dataset so it's easy for our viewers to analyze what we have
- Now, we seem to have a problem
- There are question marks in our data, which entails that these specific observations do not have values
- In this case, what do we do?

In [3]:
```
d=auto.replace('?',np.NAN)
d.isnull().sum()
```

Out[3]:
```
symboling             0
normalized-losses    41
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
dtype: int64
```

- We use the above code in order to figure out which columns from our data has question marks
- If the number is zero, we leave it alone
- If there is a number, we utilize the following code

In [4]:
```
a=auto[auto['normalized-losses']!='?']
b=(a['normalized-losses'].astype(int)).mean()
auto['normalized-losses']=auto['normalized-losses'].replace('?',b).astype(int)
```

In [5]:
```
a=auto['num-of-doors'].map({'two':2,'four':4,'?':4})
auto['num-of-doors']=a
```

In [6]:
```
a=auto[auto['horsepower']!='?']
b=(a['horsepower'].astype(int)).mean()
auto['horsepower']=auto['horsepower'].replace('?',b).astype(int)
```

In [7]:
```
a=auto[auto['bore']!='?']
b=(a['bore'].astype(float)).mean()
auto['bore']=auto['bore'].replace('?',b).astype(float)
```

In [8]:
```
a=auto[auto['stroke']!='?']
b=(a['stroke'].astype(float)).mean()
auto['stroke']=auto['stroke'].replace('?',b).astype(float)
```

In [9]:
```
a=auto[auto['peak-rpm']!='?']
b=(a['peak-rpm'].astype(float)).mean()
auto['peak-rpm']=auto['peak-rpm'].replace('?',b).astype(float)
```

In [10]:
```
a=auto[auto['price']!='?']
b=(a['price'].astype(int)).mean()
auto['price']=auto['price'].replace('?',b).astype(int)
```

- The code above will substitute the question marks with random values so nothing is left out
- Let's make some interesting analysis and visualizations with our data

How many cars/observations are in the dataset?

In [11]:
```
print('There are',auto['make'].count(),'cars/observations in the dataset')
```

```
There are 205 cars/observations in the dataset
```

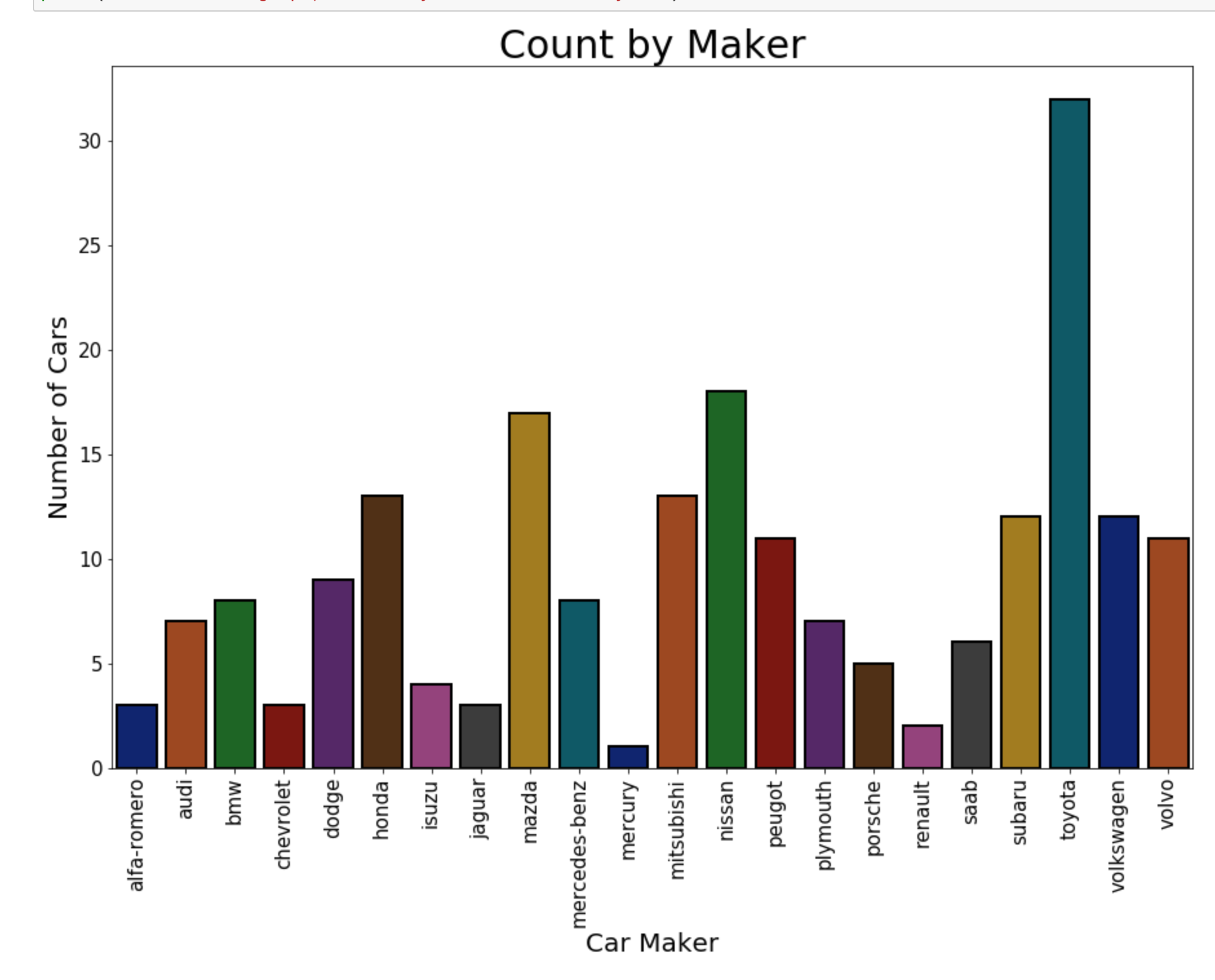
Who are the car makers in this dataset?

In [12]:
```
print(auto['make'].unique())
```

```
['alfa-romero' 'audi' 'bmw' 'chevrolet' 'dodge' 'honda' 'isuzu' 'jaguar'
 'mazda' 'mercedes-benz' 'mercury' 'mitsubishi' 'nissan' 'peugot'
 'plymouth' 'porsche' 'renault' 'saab' 'subaru' 'toyota' 'volkswagen'
 'volvo']
```

Which of the car makers in this dataset has the most cars?

In [13]:
```
fig=plt.figure(figsize=(15,10))
ax=sns.countplot(auto['make'],palette='dark',edgecolor='k',linewidth=2)
plt.xticks(rotation='vertical')
plt.xlabel('Car Maker',fontsize=20)
plt.ylabel('Number of Cars',fontsize=20)
plt.title('Count by Maker',fontsize=30)
ax.tick_params(labelsize=15)
plt.show()
print('Based on the graph, obviously the answer is Toyota')
```

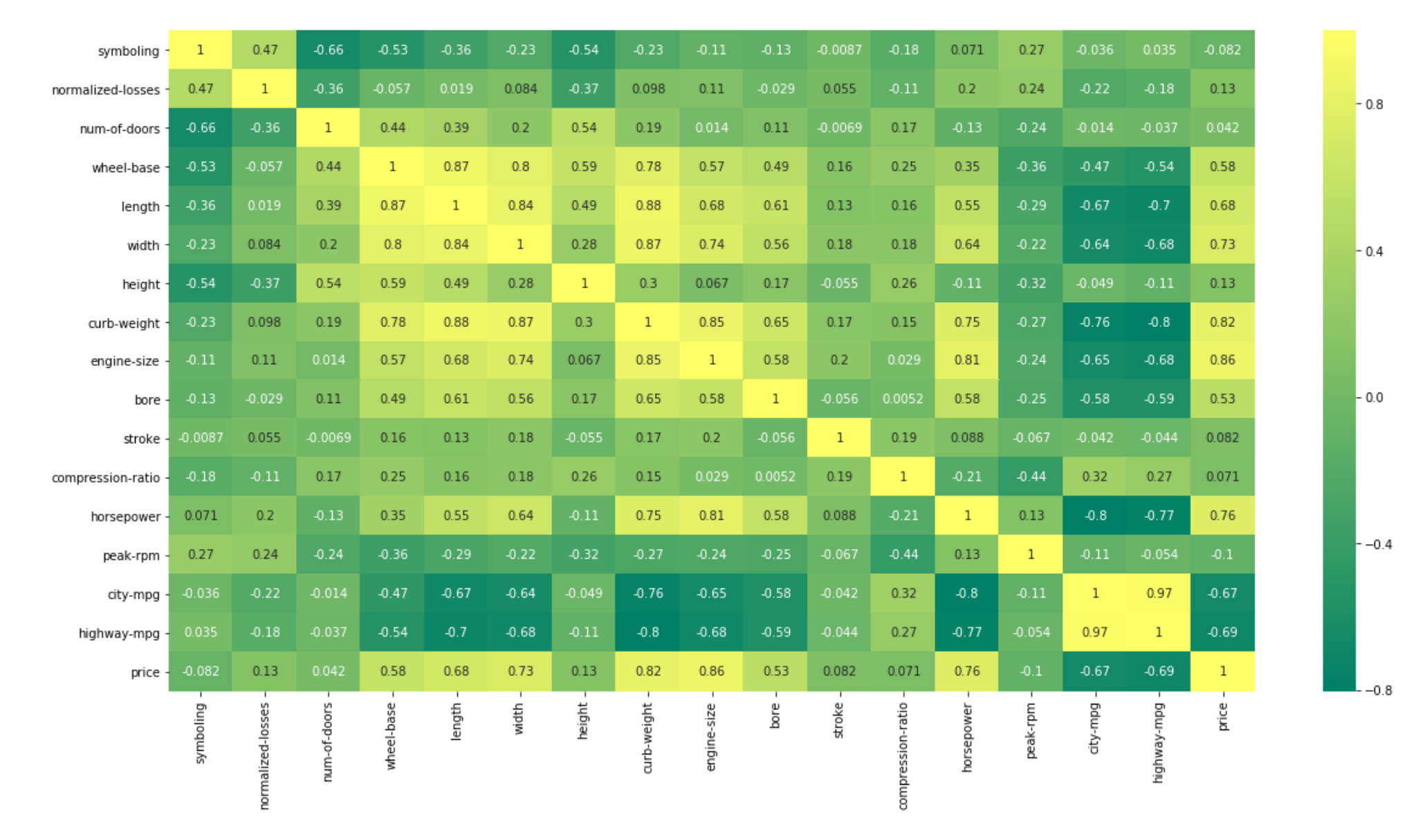



```
Based on the graph, obviously the answer is Toyota
```

- Let's move into the purpose of this dataset
- We will predict the prices of cars using machine learning concepts
- The first step is the following:

In [14]:
```
plt.figure(figsize=(20,10))
sns.heatmap(auto.corr(),annot=True,cmap='summer')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1ac29be0>

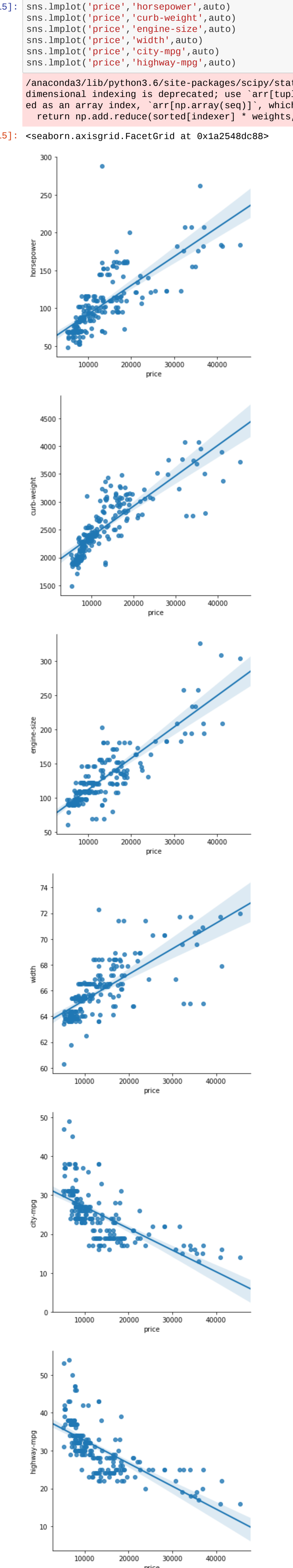


- This is a correlation heatmap, which means the relationship between two variables
- If numbers are close to zero, no relation
- If numbers are close to one, high relation
- We want to predict the prices of cars, but the way we propose on doing it is first proposing a factor analysis
- This means that by using our heatmap, we can figure out the columns in our data that are highly related to price
- Based on the heatmap, high correlations are horsepower, curbweight, engine-size, and width, whereas low correlations are city-mpg and highway-mpg
- In order to prove that, take a look at our confirmation to this

In [15]:
```
sns.lmplot('price','horsepower',auto)
sns.lmplot('price','curb-weight',auto)
sns.lmplot('price','engine-size',auto)
sns.lmplot('price','width',auto)
sns.lmplot('price','city-mpg',auto)
sns.lmplot('price','highway-mpg',auto)
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multi
dimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpret
ed as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

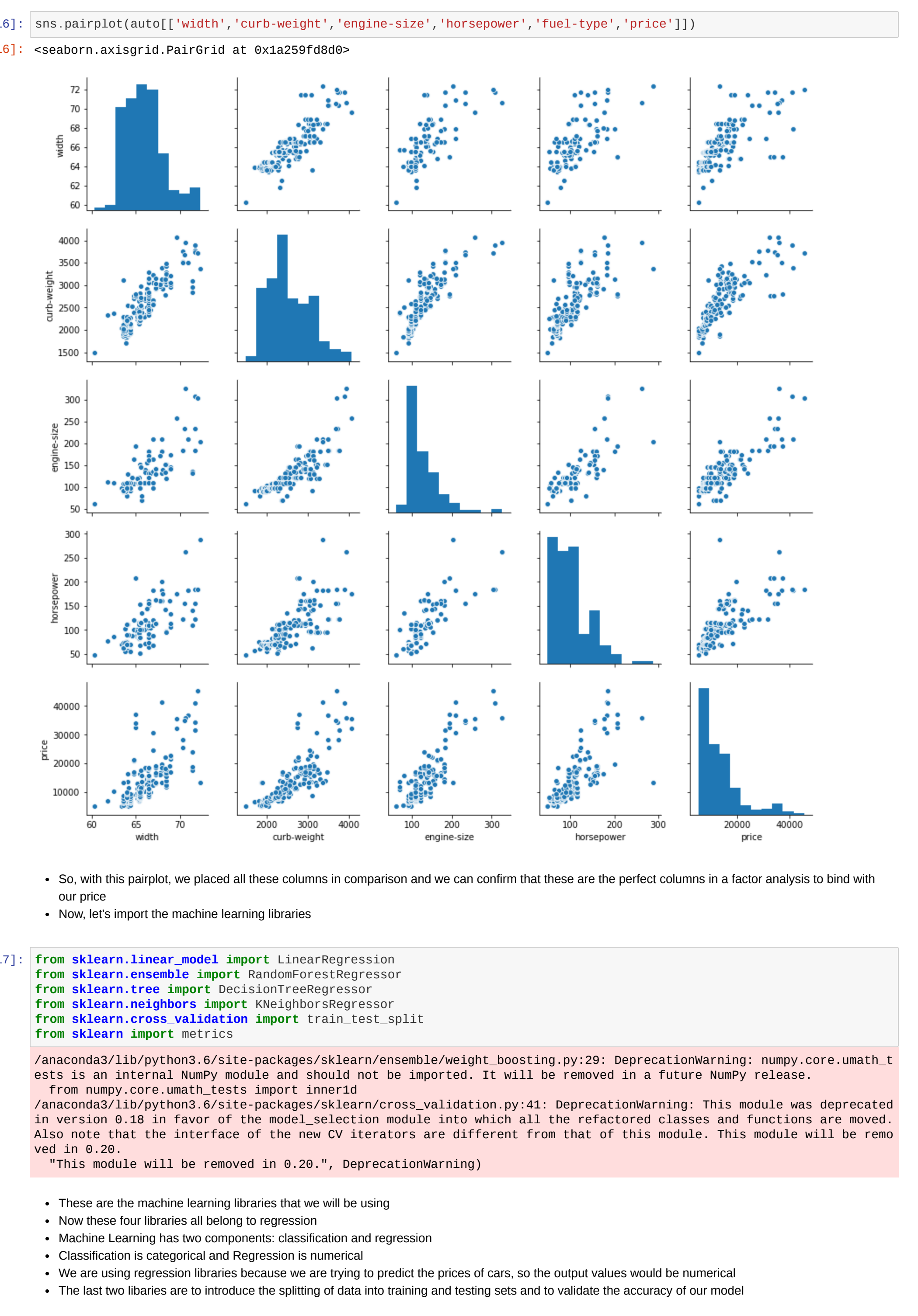
Out[15]: <seaborn.axisgrid.FacetGrid at 0x1a2548dc88>



- Based on these line plots, we can confirm that the plots are accurately summarized and can be correlated to our heatmap
- Now, let's put the columns of high correlation next to each other and see what happens

In [16]:
```
sns.pairplot(auto[['width','curb-weight','engine-size','horsepower','fuel-type','price']])
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x1a259fd8d0>



- So, with this pairplot, we placed all these columns in comparison and we can confirm that these are the perfect columns in a factor analysis to bind with our price
- Now, let's import the machine learning libraries

In [17]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.cross_validation import train_test_split
from sklearn import metrics
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.core.umath_t
ests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated
in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved.
Also note that the interface of the new CV iterators are different from that of this module. This module will be remo
ved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

- These are the machine learning libraries that we will be using
- Now these four libraries all belong to regression
- Machine Learning has two components: classification and regression
- Classification is categorical and Regression is numerical
- We are using regression libraries because we are trying to predict the prices of cars, so the output values would be numerical
- The last two libaries are to introduce the splitting of data into training and testing sets and to validate the accuracy of our model

In [18]:
```
train,test=train_test_split(auto,test_size=0.05)
X_train=train[['curb-weight','engine-size','horsepower','width']]
y_train=train.price
X_test=test[['curb-weight','engine-size','horsepower','width']]
y_test=test.price
```

- The purpose of train_test split is the following
- We utilize the training set in order for a model that can learn its data to build accuracy
- We utilize the testing set in order to test the prediction of this model on the data
- test_size of 0.05 means that we split our data 95% training and 5% testing, which is not always the case
- Usually there are splits of data such as 70-30,80-20,90-10, but the split can affect the accuracy of the model, so 95-5 was used as the best split to not alter the accuracy of our models by that much
- Our x-values are the columns or the factors that affect our price
- Our y-values are the prices
- Now, we will see the accuracy of models and make predictions

In [19]:
```
model=LinearRegression()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

Out[19]: 0.7783829385453868

In [20]:
```
model=RandomForestRegressor()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

Out[20]: 0.9778046977394528

In [21]:
```
model=DecisionTreeRegressor()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

Out[21]: 0.9970687094058951

In [22]:
```
model=KNeighborsRegressor()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

Out[22]: 0.8259453674678426

Regression Algorithms

Algorithms	Accuracy
Linear Regression	77.84%
Random Forest	97.78%
Decision Tree	99.71%
K Neighbors	82.59%

Since Decision Tree provides the highest percentage, we now do the following to obtain prediction of prices using the Decision Tree Model

In [23]:
```
model=DecisionTreeRegressor()
model.fit(X_train,y_train)
prediction=model.predict(X_test)
DTR=prediction.astype(int)
print(y_test)
```

```
20      6575
0      13495
7      18920
170    11199
161     8358
147    10198
43      6785
68     28248
84     14489
194    12940
124    12764
Name: price, dtype: int64
```

- We use the prediction model but before releasing the output, we print the array of the prices we will be predicting as shown on print(y_test)
- We see two columns, one shows the observation number and the other shows the original price of the car

In [26]:
```
print(auto.iloc[[20,0,7,170,161,147,43,68,84,194,124],2])
```

```
20         chevrolet
0        alfa-romero
7               audi
170           toyota
161           toyota
147           subaru
43             isuzu
68     mercedes-benz
84        mitsubishi
194            volvo
124         plymouth
Name: make, dtype: object
```

- We print the respective car maker present in this array
- Overall, since the arrays would be randomized, we can work on predictions using smaller arrays which are effective in this case

In [25]:
```
print(DTR)
```

```
[13207 16500 17710 13295  7198  8013 10295 25552 14869 15985 12964]
```

- These are the newly predicted array values from our most accurate model
- Here is a final analysis using the following table

Car Maker	Prices
Chevrolet	13207
Alfa-Romero	16500
Audi	17710
Toyota	13295
Toyota	7198
Subaru	8013
Isuzu	10295
Mercedes-Benz	25552
Mitsubishi	14869
Volvo	15985
Plymouth	12964

- In our output, we can clearly see that prices have been altered
- Now there may be some doubts you may have of this final output
- For example, the huge difference in the two Toyota prices
- Another example, if you wanted to run the algorithm again, but you see observations with price outputs that may not make sense to you
- If either of those examples are the case, then let me remind you that every observation and its relative price are based not only on the factors that affect price, but also on the other 20 + factors present in the original dataset
- So for example, a Toyota of 7198 may have less value than a Toyota of 13295 because of the difference in horsepower or engine-size
- Overall, this is a randomized array, so the more trials you run, the more results and possibilities you can analyze